(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 239 081 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification: 06.09.95 Bulletin 95/36

(51) Int. Cl.[6]: G06F 9/38

(21) Application number: 87104345.1

(22) Date of filing: 24.03.87

(54) Pipelined data processor capable of decoding and executing plural instructions in parallel.

(30) Priority: 26.03.86 JP 65651/86

(43) Date of publication of application: 30.09.87 Bulletin 87/40

(45) Publication of the grant of the patent: 06.09.95 Bulletin 95/36

(84) Designated Contracting States: DE FR GB

(56) References cited:
EP-A- 0 021 399
IBM TECHNICAL DISCLOSURE BULLETIN, vol. 25, no. 1, June 1982, pages 342-343, New York, US; M. LECHACZYNSKI et al.: "Central control unit instruction prefetch mechanism"
IBM TECHNICAL DISCLOSURE BULLETIN, vol. 23, no. 1, June 1980, pages 409-412, New York, US; R.E. MILLER et al.: "Floating-duplex decode and execution of instruction"
IBM TECHNICAL DISCLOSURE BULLETIN, vol. 14, no. 12, May 1972, pages 3599-3611, New York, US; B.O. BEEBE et al.: "Instruction sequencing control"
IBM TECHNICAL DISCLOSURE BULLETIN, vol. 27, no. 10B, March 1985, pages 6257-6258, New York, US; "Deferred decoding"
PATENT ABSTRACTS OF JAPAN, vol. 6, no. 237 (P-157)[1115], 25th November 1982; & JP-A-57 137 944 (FUJITSU K.K.) 25-08-1982

(73) Proprietor: HITACHI, LTD.
6, Kanda Surugadai 4-chome
Chiyoda-ku, Tokyo 101 (JP)

(72) Inventor: Kuriyama, Kazunori Hitachi Dai-2 Kyoshinryo
1-3, Higashikoigakubo-3-chome
Kokubunji-shi (JP)
Inventor: Shintani, Yooichi Hitachi Tennoue Shataku B13
217, Higashikoigakubo-1-chome
Kokubunji-shi (JP)
Inventor: Yamaoka, Akira Hitachi Koyasudai Apartment B-203
32, Koyasumachi-2-chome
Hachioji-shi (JP)
Inventor: Shonai, Tohru
1, Horiyamashita
Hadano-shi (JP)
Inventor: Kamada, Eiki Hitachi Koyasudai Apartment D-201
32, Koyasumachi-2-chome
Hachioji-shi (JP)
Inventor: Inoue, Kiyoshi 11,
Senkawamachi-1-chome
Toshima-ku
Tokyo (JP)

(74) Representative: Strehl Schübel-Hopf Groening & Partner
Maximilianstrasse 54
D-80538 München (DE)

EP 0 239 081 B1

Description

BACKGROUND OF THE INVENTION

The present invention relates to a data processor for executing instruction in a pipelined and parallel fashion, and more particularly to a data processor which simultaneously decodes two instructions and executes them in a parallel and pipelined fashion.

In a prior art large scale general purpose computer, instructions are executed in a pipelined fashion to execute different instruction in an overlapped manner so that an execution speed of instruction sequence is effectively increased. In order to improve it, various techniques have been proposed. For example, in JP-A-60-17538, in order to execute an instruction which requests operations for two memory operands, two address adders are provided so that addresses of the two memory operands are simultaneously calculated. In JP-A-58-176751, two pipelined instruction execution units simultaneously (or parallelly) decode two instructions in one instruction sequence to parallelly execute them. In JP-A-59-32045 (or corresponding U.S. Patent 4,626,989), in order to improve the above parallel decode technique, if the two instructions cannot be executed in parallel because a result of execution of a first instruction to be executed first is used by a second instruction to be executed later, the second instruction is executed following to the first instruction in the same pipelined instruction execution unit as that for the first instruction.

In the prior art parallel decode technique, each of the two pipelined instruction execution units needs an address generator, an operand read memory and an operation unit. When a data processor is constructed, the operation units and other units are provided in deplicate. Thus, investment of hardwares is large and a control configuration is complex. Since the operation unit comprises a plurality of units which perform various operations, a circuit scale thereof is large.

IBM Technical Disclosure Bulletin, vol. 23, no. 1, June 1980, pages 409-412; "Floating-duplex decode and execution of instruction" discloses a pipeline data processor comprising only one operation unit and executing two consecutive instructions in parallel provided that they are data-independent, the first one is not a branch instruction and only one needs an operation.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a data processor whose hardware scale is not large to compare with that of a pipelined instruction execution unit in a prior art data processor and which can decode a plurality of instructions in parallel.

The above object is achieved by means as set out in claim 1.

The instructions to be executed include an instruction which uses the operation unit as well as an instruction which used a resource other than the operation unit. When the detection means detects that those two instructions have been extracted into the pair of registers, those two instructions are executed in parallel by the instruction execution means. As a result, the two instructions which use the operation unit and other resource are executed in parallel. Thus, two instructions can be executed in parallel without providing two operation units.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows an overall configuration of a processor which embodies the present invention,
Figs. 2a and 2b show a detail of an instruction extraction circuit 3 of Fig. 1,
Figs. 3a, 3b and 3c show various instruction formats,
Fig. 4 shows a detail of a selector 4 of Fig. 1 for selecting a register number used to generate an address,
Fig. 5 shows a circuit for generating a select signal and a memory read request, of a decode control circuit 6 of Fig. 1,
Fig. 6 shows a detail of an instruction read adder, of the decode control circuit 6 of Fig. 1,
Fig. 7 shows a detail of a conflict detection circuit 5 of Fig. 1,
Figs. 8a, 8b, 8c and 8d show an instruction execution flow by the present invention,
Fig. 9 shows an overall configuration of another processor which embodies the present invention, and
Fig. 10 shows a detail of a selector 800 of Fig. 9.
Fig. 11 shows a detail of a decode control circuit 803 of Fig. 9.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Embodiments of the present invention will be explained with reference to the drawings.

Fig. 1 shows an overall configuration of a pipelined data processor which utilizes the present invention. The data processor comprises two instruction registers 1 and 2, an instruction extract circuit 3 for simultaneously extracting two instructions in an instruction buffer 99 into those instruction registers 1 and 2, a selector 4 for selecting a general purpose register number in a group of instruction registers and supplying it to a group of general purpose registers 7, two address generation circuits 8 and 9 for generating memory data addresses based on data read from the group of general purpose registers, a main memory

13, an operand buffer memory 10 and instruction buffer memory 11 accessed by addresses generated by the address generation circuits 8 and 9, an operation unit 12 for operating an operand read from the buffer memory 10 and storing a result into the buffer memory 10 or the general purpose register 7, a decode control circuit 6 for decoding instructions in the instruction registers 1 and 2 to control the execution thereof, and a conflict detection circuit 5, a selector 90 for selecting an instruction code and a register number of a register operand from instruction registers which hold instructions utilizing an operation unit 12, among the instructions extracted and set in the instruction registers 1 and 2, and registers 89 for holding the selected instruction code and the selected register number until the start of operation in timed relation to each pipelined stage. The operation unit 12 comprises a plurality of operation circuits for performing arithmetic or logical operations required by the instructions executed by the processor. The control of operation is performed by supplying via a line 181 to the operation unit 12 an instruction code held in the registers 89 in time relation to each pipelined stage.

The present processor sequentially executes the instructions in a pipelined fashion. The instruction buffer memory 11 is provided separately from the operand buffer memory 10 so that when the memory 10 is used for the execution of an instruction, a subsequent instruction is fetched in parallel.

The length of instruction executable by the present processor is 2, 4 or 6 bytes.

Fig. 3a shows an instruction format of a four-byte load instruction. The load instruction reads an operand data in the memory and stores it into the general purpose register. In Fig. 3a, OP represents an operation code, R1 represents a general purpose register number in which data read from the memory is to be stored, X and B represent two general purpose register (index register and base register) numbers for calculating an operand address in the memory, and D represents a displacement. The operand address is calculated by summing a content of the index register (X), a content of the base register (B) and the displacement D.

Fig. 3b shows an instruction format of a four-byte branch instruction. The branch instruction compares a mask value (M1) of the branch instruction and a condition code to determine whether to branch or not. A destination instruction address is determined by X, B and D of the instruction. The meanings of the bits of the instruction are identical to those of the load instruction, except for the mask value (M1).

Fig. 3c shows a six-byte instruction. It holds two operands to be operated in the memory. To start the operation, an address of a first operand is calculated based on B1 and D1, then an address of a second operand is calculated based on B2 and D2 in order to read the operands.

An example of a two-byte instruction is not shown for simplification purpose.

Specific specifications of detail of the instructions are described in HITAC manual "M Series Processor (M/EA mode)" (No. 8080-2-083 (1984) pp 12-14, 94, 179 and 195-198).

An outline of the operation of the present processor is now explained.

The instruction registers 1 and 2 have 6-byte and 4-byte lengths, respectively. Instructions (8 byte length) are sequentially fetched from the instruction buffer memory 11 through a signal line 132 and stored into the instruction buffer 99. The instruction extract circuit 3 extracts instructions (8-byte length) including the instruction to be next executed, of the instructions stored in the instruction buffer 99. When the instruction to be next executed is of 2-byte or 4-byte length, two instructions are included in the extracted instructions, and the instruction extract circuit 3 sets the instruction to be next executed (first instruction) and the following instruction (second instruction) into the instruction registers 1 and 2, respectively. When the instruction to be next executed is of 6-byte length, only the first instruction is set into the instruction register 1.

The operation when the first and second instructions are set into the first and second instruction registers is explained.

In the present invention, when there is no conflict of register between the first and second instructions or conflict between the resources (address generation circuits 8, 9 and buffer memories 10, 11), both instructions are simultaneously executed, and if there is any conflict, only the first instruction is preferentially executed. The necessary processing therefor is carried out by the conflict detection circuit 5 and an instruction combination recognition circuit 96. The former detects a conflict that a general purpose register in which a result of execution of the first instruction is to be written is designated by the second instruction as the index or base register. The latter detects a conflict that the first and second instructions use the same address generation circuit 8 or 9.

When the first instruction requires the fetching of the memory operand and the use of the operation unit and the second instruction reguires the fetching of the instruction and there is no register conflict between those instructions, those instructions can be executed in parallel in the present embodiment because the operand buffer 10 and the instruction buffer 11 are separately provided. The selector 4 selects an index register number, a base register number and a displacement specified by the first instruction, and reads the contents of the general purpose registers 7 corresponding to those register numbers. The address generation circuit 8 (which is a three-input adder) adds the contents of the registers and the displacement to generate an address for memory ac-

cessing. The generated address is sent to the buffer memory 10 and used to fetch the operand specified by the first instruction.

The selector 4 selects an index register number, a base register number and a displacement specified by the second instruction, and reads the contents of the general purpose registers 7 corresponding to those register numbers. The address generation circuit 9 (which is also a three-input adder) adds the contents of the general purpose registers and the displacement to generate an address for memory accessing. The generated address is sent to a buffer memory 11 and used to fetch the instruction specified by the second instruction.

When the first instruction requires the fetching of the instruction and the second instruction requires the fetching of the memory operand and the use of the operation unit, those instructions are executed in parallel. The address generation circuit 8 and the buffer memory 10 are used for the instruction 2, and the address generation circuit 9 and the buffer memory 11 are used for the instruction 1.

When both the first instruction and the second instruction require the fetching of the memory operands, the two instructions cannot be executed in parallel because there is only one operand buffer 10 and only one operation unit 12 in the present embodiment, and only the first instruction is executed. The address generation circuit 8 and the buffer memory 10 are used.

When both the first and second instructions require the fetching of the instructions, only the first instruction is executed. The address generation circuit 9 and the buffer memory 11 are used for the instruction 1.

When only the instruction 1 is executed, the instruction extract circuit 3 extracts a group of instructions (8-byte length) which include the instruction 2 as an instruction to be next executed, from the instruction buffer 99, and the instruction 2 is set into the instruction register 1 and the instruction to be next executed is set into the instruction register 2. Thus, the instruction 2 is executed one cycle behind the instruction 1.

The instruction code and register number of an instruction allowed to execute are selected by the selector 90 and delivered to the registers 89. The registers 89 comprise a plurality of registers connected in series. The content of each register is successively delivered to the next register. An instruction is executed by a plurality of pipelined stages, with each stage corresponding to each of registers. The content of each register is used to control the corresponding stage. When an instruction allowed to execute utilizes data in the general purpose registers 7 as an operation operand, the register numbers R1 and R2 specified by the instruction are delivered via a line 180 to the general purpose registers 7. An operand is read out on a line 182 and delivered to the operation unit 12.

The decode control circuit 6 includes a selector control circuit 94 which controls the selector 4, a request generation circuit 94 which issues a fetch request to the buffer memory 10 or 11, and an end of decode detection circuit 99 which controls the entire circuit.

Detail of the processor of Fig. 1 is now explained. Detail of the instruction extract circuit 3 is shown in Figs. 2a and 2b, in which Fig. 2a shows a hardware configuration and Fig. 2b illustrates an operation thereof. A signal line 199 denotes 8-byte instructions fetched from the instruction buffer 99, and a signal line 150 denotes first two bits of the eight bytes. The instructions handled in the present processor have an instruction format of the HITAC-M Series computer, and the length of the instruction is specified by the first two bits of the instruction. When the first two bits of the instruction are "00", the instruction is of 2-byte length, when they are "11", it is of 6-byte length, and when they are neither "00" nor "11", it is of 4-byte length. (Detail of the M Series instruction specification is described in HITAC manual "M Series Processor (M/EA mode)" (No. 8080-2-083 (1984) pp 63-64).) The decoder 158 decodes the first two bits to control the selector 159, and produces first and second valid signals V1 and V2 indicating whether valid instructions have been sent to the first and second instruction registers 1 and 2. For example, when the first two bits of the instruction is "00", the instruction is of 2-byte length and the selector 159 extracts the first and second bytes of the eight bytes on the signal line 199 onto the signal line 101 and sends them to the first instruction register. On the other hand, in order to extract the instruction next to the instruction extracted onto the signal line 101 from the eight bytes on the signal line 199, the selector 159 extracts the third to sixth (four) bytes onto the signal line 102 and sends them to the second instruction register. When the first two bits of the instruction indicate the four-byte length of the instruction, the bytes 1 - 4 on the signal line 199 are extracted onto the first instruction register 1, and the next instruction, that is, the bytes 5 - 8 are extracted from the signal line 199 into the second instruction register 2. When the first instruction is of 6-byte length, the bytes 1 - 6 on the signal line 199 are extracted and sent to the first instruction register 1, and nothing is sent to the second instruction register 2. Namely, when the 6-byte length instruction is extracted into the first instruction register, the immediately following instruction is not executed in parallel.

The decoder generates "1" on V1 and V2 when the first two bits on the line 199 is "00", "01" or "10", and generates "0" on V2 when the first two bits are "11" to indicate that a valid instruction has not been sent to the second instruction register 2. It should be

noted that when the instruction set in the second instruction register 2 is of 6-byte length, all bytes are not aligned even if the signal V2 is set on the line 154.

The instructions fetched from the instruction buffer 99 are extracted by the instruction extract circuit 3 into the instruction registers 1 and 2 as the first and second instructions. Let us assume that a load instruction is extracted into the instruction register 1 and a branch instruction is extracted into the instruction register 2.

When the load instruction and the branch instruction are extracted into the instruction registers 1 and 2, respectively, the load instruction refers the operand buffer memory 10 to fetch the operand, and the branch instruction refers the instruction buffer memory 11 to fetch a branch-to instruction. This operation is explained below.

When the instructions are extracted into the instruction registers 1 and 2, the operation codes of the first and second instructions are decoded by the decoder 98 in the decode control circuit 6, possibility of parallel execution of the two instructions is checked by the instruction combination recognition circuit 96, a request signal to the memory is generated by the request generation circuit 94, and a control signal to the selector 4 is generated by the selector control circuit 95. When the end of instruction decoding is detected by the end of decode detection circuit 97, the address in the instruction fetch address generation circuit 93 is incremented by the instruction length to fetch the next instruction.

Detail of the selector 4 is shown in Fig. 4. Numerals 190, 193 - 195 denote two-input, one-output selectors, and numerals 191 and 192 denote three-input, one-output selectors. The selectors 190 - 195 select the register numbers of the index register and the base register or the displacements and supply them to the general purpose register 7 and the address generation circuits 8 and 9. The select operation is controlled by the signal line 117 as follows. For example, the index register numbers of the first instruction and the second instruction are supplied to the selector 190. When the control line 269 is not set, the index register number 106 of the first instruction is selected and supplied to the signal line 119. The data read by using this register number is supplied to the address generation circuit 8 through the signal line 124. If the decode control circuit 6 determines that the address calculation of the second instruction be conducted by the address generation circuit 8, the signal line 269 is set, and the index register number 112 of the second instruction is selected and supplied to the signal line 119.

In the selector 191, when the control signals 270 and 265 are not set, the base register number 107 of the first instruction is selected and supplied to the signal line 120, and if the decode control circuit 6 determines that the address calculation of the second instruction be conducted by the address generation circuit 8, the control line 270 is set and the base register number 113 of the second instruction is selected. If the decode control circuit 6 determines that the address calculation of the 6-byte length (SS format) second instruction be conducted, the signal line 109 is selected. Since the 6-byte instruction has two memory operands, two times of memory reference are required in the present processor, and the signal line 109 if selected for the memory reference of the second operand. Other selectors control in a similar manner. The selector 192 selects the displacement 114 if the signal line 271 is set, and selects the displacement 110 if the signal line 265 is set. The selector 193 selects the index register number 112 if the signal line 266 is set, the selector 194 selects the base register number 113 if the signal line 267 is set, and the selector 195 selects the displacement 114 if the signal line 268 is set.

The register numbers on the signal lines 119 - 122 of the outputs of the selector 4 are supplied to the general purpose register 7 and used for fetching data. The content of the register specified by the signal line 119 is read onto the signal line 124. Similarly, the contents of the registers specified by the signal lines 120, 121 and 122 are read onto the signal lines 125, 126 and 127, respectively. It is now assumed that the load instruction is set in the instruction register 1 and the branch instruction is set in the instruction register 2. Thus, the content of the index register specified by the load instruction is read onto the signal line 124, the content of the base register specified by the load instruction is read onto the signal line 125, the content of the index register specified by the branch instruction is read onto the signal line 126, and the content of the base register specified by the branch instruction is read onto the signal line 127, through the selector 4 and the general purpose register 7. None of signal lines 117 is not set at this time. The address generation circuit 8 receives the data on the signal lines 124 and 125 and the displacement on the signal line 118 and adds them, and transfers the resulting address to the operand buffer memory 10 through the signal line 128, transfers the operand data fetched therefrom to the signal line 129 and the operation unit 12 and to the general purpose register 7 through the signal line 130 without any operation in the operation unit 12 and stores it in the general purpose register specified by the first instruction. On the other hand, the address generation circuit 9 receives the data on the signal lines 126 and 127 and the displacement on the signal line 123 and adds them to calculate a branch-to address of the branch instruction. The sum is transferred to the instruction buffer memory 11 through the signal line 131 to fetch the branch-to instruction. The branch-to instruction is required when the branch instruction successes the branch. In this case, the branch-to instruction fetched from the buf-

fer memory 11 is supplied to the instruction buffer 99 through the signal line 132 and extracted to the instruction registers 1 and 2.

The decode control circuit 6 comprises the instruction combination recognition circuit 96 which controls the selector 4 depending on the combination of the instructions extracted into the instruction registers 1 and 2, the selector control circuit 95, the request generation circuit 94 for issuing a memory reference request to the buffer memory 10 or 11, the end of decode detection circuit 97 for detecting the end of decoding of the instruction, and the instruction fetch address generation circuit for updating the instruction fetch address.

Figs. 5 and 6 show details of the decode control circuit 6. In Fig. 5, numeral 98 denotes a decoder which decodes the operation codes OP1 and OP2 of the first and second instructions and reflects the results to the signal lines 254 - 263. Numeral 203 denotes a flip-flop which has a function to delay the input signal one cycle. Numerals 200 and 209 denote inverter gates which logically invert the input signals. Others are AND or OR gates.

A signal DS on a signal line 251 indicates the end of decoding of the instruction. It is set when the signal V1 indicating that the first instruction is valid is set on the signal line 153, or when the signal V2 indicating that the second instruction is valid is set on the signal line 154 and a conflict signal CONF is not set on the signal line 115. This circuit utilizes the fact that if the second instruction is valid, the first instruction is always valid. When the coding of the first and second instructions is completed by the signal line 251, the instruction fetch address is incremented by an instruction fetch address generation circuit 93 (Fig. 6) so that the succeeding instructions are extracted into the instruction registers 1 and 2. In Fig. 6, a signal line 133 denotes a fetch address of the instruction to be fetched from the instruction buffer memory 11. When the decoding of the instruction is completed, the address of the previous instruction in the register 232 is incremented by the instruction length 284 of the decoded instruction by the adder 231 and the result is held in the register 232 for use in fetching the next instruction from the buffer memory 11. The end of instruction decode signal line 251 of Fig. 5 is used to set the register 232.

The instruction length of the decoded instruction is supplied to the signal line 284. The signal line 259 is set when the first instruction uses the address generation circuit 8, and the signal line 260 is set when the second instruction uses the address generation circuit 8. The signal line 255 is set when the first instruction uses the address generation circuit 9, and the signal line 274 is set when the second instruction uses the address generation circuit 9. The signal lines 153 and 154 indicate the validity of the first and second instructions, respectively. The signal line 281 is set when the first and second instructions cannot be simultaneously decoded by the conflict of the address generation circuits 8 and 9. The signal line 254 is set when the first instruction is of 6-byte length, and the signal line 280 is set when the second instruction is of 6-byte length. As described above, when the first instruction is of 6-byte length, the second instruction is not decoded, and when the second instruction is of 6-byte length, the second instruction is not decoded in parallel. As a result, the signal line 286 is set only when the decoding of only the first instruction is completed and the succeeding instruction is to be decoded in the next decode cycle. Thus, in this case, the instruction fetch address is incremented by the instruction length IL1 of the first instruction. When the signal line 286 is not set, the first and second instructions can be decoded in parallel. In this case, the address is incremented by a sum of the instruction length of the first instruction and the instruction length of the second instruction (IL1 + IL2). Thus, the instruction length IL1 of the first instruction is set in the signal line 282, and the sum IL1 + IL2 of the instruction length IL1 of the first instruction and the instruction length IL2 of the second instruction is set in the signal line 283 so that the instruction length of the decoded instruction is set in the signal line 284 and the next instruction fetch address is produced on the line 285 by adding the original instruction address in the register 32 and the instruction length on the line 284 by the adder 231. The next instruction fetch address is set in the register 232.

If the decoded instruction is the branch instruction, not the original instruction address in the register 232 but the branch-to instruction address 131 generated by the address generation circuit 9 of Fig. 1 is selected by the selector 290 and added to the instruction length on the line 284.

In Fig. 5, the signal line 265 is set when the first instruction is of 6-byte length and the second operand of the instruction is to be fetched. It is set in the second cycle from the extraction of the 6-byte instruction into the instruction register 1. The signal line 253 indicates one-cycle delay of the end of decode signal, and the signal line 254 is set when the first instruction is of 6-byte length and meets the condition for the signal line 265.

The signal line SEL1XAA9 266 is set when the first instruction uses the address generation circuit 9 and requires to read the index register. The signal line 255 is set when the first instruction uses the address generation circuit 9, and the signal line 256 is set when the first instruction requires to read the index register. Since the signal line 257 is set when the first instruction requires to read the base register, the signal line SEL1BAA9 267 is set when the first instruction uses the address generation circuit 9 and requires to read the base register. Similarly, the signal line SEL1DAA9 268 is set when the first instruction

uses the address generation circuit 9 and requires the addition of the displacement.

Since the signal line 261 is set when the second instruction requires to read the index register, the signal line SEL2XAA8 269 is set when the second instruction uses the address generation circuit 8 and requires to read the index register. Similarly, since the signal line 262 is set when the second instruction requires to read the base register, the signal line SEL2BAA8 270 is set when the second instruction uses the address generation circuit 8 and requires to read the base register. Since the signal line 263 is set when the second instruction requires the addition of the displacement, the signal line SEL2DAA8 271 is set when the second instruction uses the address generation circuit 8 and requires the addition of the displacement. The address calculation by the second instruction cannot always be performed and the possibility of execution is examined by the instruction combination recognition circuit 96. In this circuit, the signal line 259 is set when the first instruction uses the address generation circuit 8, and the signal line 260 is set when the first instruction uses the address generation circuit 8. Thus, the signal 264 indicates the condition to permit the use of the address generation circuit 8 by the second instruction. The signal lines 265 - 271 have been explained in connection with Fig. 4.

As seen from the above description, the signal line 259 is set when the first instruction uses the address generation circuit 8, the signal line 264 is set when the second instruction uses the address generation circuit 8, and the signal line 265 is set when the first instruction is of 6-byte length and the address calculation of the second operand of the instruction is to be carried out by the address generation circuit 8. The operand fetch request REQ 10 on line 272 for the operand memory 10 may be set in the above three cases. On the other hand, since the signal line 255 is set when the first instruction uses the address generation circuit 9 and the signal line 274 is set when the second instruction uses the address generation circuit 9, the instruction fetch request REQ 11 to the instruction buffer memory 11 may be set in the above two cases.

In this manner, the decode control circuit 6 decodes the operation codes of the first and second instructions to effect control necessary to the memory reference of the first instruction (load instruction), that is, issues input data select control of the address generation circuit 8 and memory read request to the buffer memory 10, and effects control necessary to memory reference of the second instruction (branch instruction), that is, issues input data select control of the address generation circuit 9 and memory read request to the buffer memory 11. It also detects the end of decoding of the first and second instructions to exactly calculate the addresses of the instructions to be next extracted into the instruction registers 1 and 2 by using the instruction address adder 231 so that the instructions are sequentially fetched.

In the above description, it has been assumed that the load instruction and the branch instruction extracted into the instruction registers 1 and 2 can always refer the memory in parallel and the decoded in parallel. However, in the following case, the second instruction (branch instruction) cannot refer the memory in parallel with the load instruction. Namely, when the register number specified by the index register or the base register of the branch instruction is specified by an R1 part of the first instruction (load instruction) for updating the content thereof, it is necessary to generate the branch-to instruction address of the second instruction (branch instruction) after the operand is fetched from the buffer memory 10 by the first instruction (load instruction).

Detail of the conflict detection circuit 5 which detect whether the first instruction changes the register used in the address generation of the second instruction when the instructions are extracted into the instruction registers 1 and 2 is shown in Fig. 7. The signal line 105 indicates a register number GR1 to be updated by the first instruction, and the signal lines 112 and 113 indicate the index register number 2X and the base register number 2B used in the address generation by the second instruction. They are compared by comparators 300 and 301, and if they match, the signal lines 313 and 314 are set. Those signals are logically ANDed by gates 302 and 303 with the signal line CHG1 310 which indicates that the first instruction changes the content of the general purpose register specified by R1 and the signal line NEED2X 312 which indicates that the second instruction requires the readout of the index register or the signal line NEED2B which indicates that the second instruction requires the readout of the base register. If the register number to be changed by the first instruction is equal to the register number to be read by the second instruction, the signal line CONF 115 is set.

The signal line 115 is connected to the decode control circuit 6. In Fig. 5, if the signal line 115 is set, the AND gate 201 is not conditioned and the AND gates 210 and 217 which use the output signal 250 of the AND gate 201 are also not conditioned so that the read request to the memory by the second instruction is suppressed. In Fig. 6, when the signal line 115 is set, the output 286 of the OR gate 226 is set and the instruction length 282 of the first instruction is selected and it is supplied to the next instruction fetch address calculation adder 231 through the signal line 284 as an increment.

If the conflict detection circuit 5 detects that the register to be updated by the first instruction is to be read as the index or base register of the second instruction, the end of decode detection circuit 99 controls only the execution of the first instruction, and the

second instruction is again extracted into the instruction register 1 by the instruction fetch address generation circuit after the first instruction has been decoded. It is necessary to suppress the decoding of the branch instruction until the preceding load instruction operand is fetched. This may be done by a conventional logic and is not explained here.

The operation performed when the branch instruction and the load instruction are extracted into the instruction register 1 and the instruction register 2, respectively, is now explained. In Fig. 5, since the first instruction uses the address generation circuit 9, the signal line 255 is set, and the select signals 266 - 268 are set. Since the second instruction uses the address generation circuit 8, the signal line 260 is set and the signal line 259 is not set. Accordingly, the signal line 264 is set and the select signals 269 - 271 are set. As a result, the data 112 - 114 of the second instruction (load instruction are selected by the selectors 190 - 192 of Fig. 4 and supplied to the signal lines 118 - 120. On the other hand, the data 106 - 108 of the first instruction (branch instruction) are selected by the selectors 193 - 195 and supplied to the signal lines 121 - 123. In this manner, the address generation circuit 8 calculates the operand address of the second instruction (load instruction), and the address generation circuit 9 calculates the branch-to address of the first instruction (branch instruction).

A read request 272 to the buffer memory 10 and a read request to the buffer memory 11 are set, and the memories are referred by using the addresses calculated above.

In Fig. 6, since none of the signal lines 115, 254, 255, 259, 260, 274 and 280 is set, the output of the OR gate 226 is not set and the sum (IL1 + IL2) of the first instruction length IL1 and the second instruction length IL2 is selected by the AND gate 228 and supplied to the adder 231. As a result, the instruction fetch address is incremented by the length of the two instructions, the branch instruction and the load instruction.

In this manner, whichever of the instruction registers 1 and 2 the instruction which uses the address generation circuit 8 or 9 is extracted to, the exact corresponding address generation circuit is selected and the address calculation and the memory reference are performed.

As described above, when the two instructions which do not cause conflict between the address generation circuits 8 and 9 are extracted into the instruction registers 1 and 2, only the conflict between the registers need be detected. However, depending on the combination of the two instructions, it is necessary to suppress parallel decoding of the two instructions in the address generation stage. This is a feature of the present invention and will be specifically described below.

An operation performed when load instructions are extracted into both the instruction registers 1 and 2 is first explained, and then an operation performed when branch instructions are extracted into both the instruction registers 1 and 2 is explained.

When the load instructions are extracted into the instruction registers 1 and 2, the instruction combination recognition circuit 96 of Fig. 5 sets the signal line 259 to indicate that the first instruction uses the address generation circuit 8, and the output signal of the inverter gate 209 assumes logical "0". Accordingly, even if the signal 260 is set to indicate that the second instruction uses the address generation circuit 8, the AND gate 210 is not conditioned and a use permission signal 264 for the use of the address adder 8 by the second instruction is not set. Accordingly, the output signals of the AND gates 211 - 213 are not set. Therefore, in the selectors 190 - 192 of Fig. 4, the data lines 106 - 108 of the first instruction are selected and supplied to the signal lines 118 - 120. As a result, the address generation circuit 8 can calculate the operand address of the load instruction in the instruction register 1. In Fig. 5, when the signal line 259 is set, the operand fetch request signal 272 to the buffer memory 10 is set and the operand is fetched from the buffer memory 10. On the other hand, since none of the first and second instructions uses the address generation circuit 9, the signal lines 255 and 274 are not set and the signal line 273 is not set. As a result, the buffer memory 11 is not read. In the instruction combination recognition circuit 96II of Fig. 6, since the signal lines 259 and 260 which indicate that the first and second instructions use the address generation circuit 8 are set, the output of the AND gate 220 is set and the signal line 286 is set through the signal line 281. As a result, the instruction length IL1 282 of the first instruction is selected and supplied to the adder 231. The instruction fetch address is incremented by the length of the load instruction in the instruction register 1 and the load instruction in the second instruction register 2 is extracted into the instruction register 1.

In this manner, when the load instructions are extracted into the instruction registers 1 and 2, the instruction combination recognition circuit 96I and 96II controls such that only the load instruction in the instruction register 1 is executed and the load instruction in the instruction register 2 is executed one cycle later.

The operation performed when the branch instructions are extracted into the instruction registers 1 and 2 is now explained. Since both the first and second instructions use the address generation circuit 9, the decode signals 255 and 274 of Fig. 5 are set. Since the branch instruction does not use the address generation circuit 8, the instruction combination recognition circuit 96I is not activated. As a result, the select signals 266 - 268 and the read request 273 to the buffer memory 11 are set. In the selectors 193

- 195 of Fig. 4, the data 106 - 108 of the first instruction (branch instruction) are selected and supplied to the signal lines 121 - 123. Thus, the branch-to instruction address of the first instruction is calculated in the address generation circuit 9. Since the signal line 273 has been set, the branch instruction can be fetched from the buffer memory 11. Since none of the first and second instructions uses the address generation circuit 8, the read request 272 to the buffer memory 10 is not set. In the instruction combination recognition circuit 96II of Fig. 6, since the AND gate 221 is comditioned, the instruction length 282 of the first instruction is selected and supplied to the adder 231. The instruction fetch address is incremented by the length of the branch instruction in the instruction register 1, and the branch instruction in the instruction register 2 is again extracted into the instruction register 1.

As explained above, in the processor shown in Fig. 1, the two instructions which are consecutive on the main memory 13 are fetched from the buffer memory 11 in parallel and extracted into the instruction registers 1 and 2. The conflict detection circuit 5 detects the conflict between the registers that the general purpose register to be updated by the first instruction is to be read by the second instruction during the address generation, and the instruction combination recognition circuit 96 detects the conflict that the first and second instructions use the same address generation circuit or buffer memory. In this manner, the parallel execution of the first and second instructions is enabled and the execution time is shortened.

An effect of the high speed operation in the present processor is explained with reference to Fig 8, in which D, A, L, E and P denote instruction execution cycles in a pipeline system. In the cycle D, an instruction is decoded and an address is generated, in the cycle A, the buffer memory is referred, in the cycle L, the data read from the memory is transferred, in the cycle E, the data is operated, and in the cycle P, the result is stored. A branch instruction does not requlre the operation in the cycle E and the storing in the cycle P.

In Figs. 8a and 8b, (i) is a load instruction, (ii) is a branch instruction, and (iii) is a branch-to instruction of the branch instruction. In a prior art processor in which plural instructions are not decoded in parallel, the load instruction and the branch instruction are decoded serially in each cycle as shown in Fig. 8a. Thus, the decoding of the branch-to instruction can be started two cycles after the end of the decoding (cycle D) of the branch instruction (ii). On the other hand, in the present invention, since the load instruction and the branch instruction can be decoded in parallel, the branch instruction (ii) can be decoded in parallel with the load instruction (i) as shown in Fig. 8b, and the operand fetching of the load instruction and the fetching of the branch-to instruction of the branch instruction can be executed in parallel. Thus, the decoding of the branch-to instruction is attained one cycle earlier than that in Fig. 8a. In Fig. 8b, the execution cycles for the branch instruction (ii) are designated by D', A', L', E', and P'. In Fig. 8b, the operation cycle E of the load instruction and the operation cycle E' of the branch instruction are performed in parallel, but only one operation unit is required because the branch instruction does not need operation.

In Figs. 8c and 8d, (i) is a branch instruction, (ii) is an add instruction, and (iii) is an instruction which uses in the address calculation a general purpose register modified by the add instruction. An instruction format of the add instruction is same as that shown in Fig. 3a. Let us assume that the branch instruction (i) fails to branch. In the prior art processor in which plural instructions are not decoded in parallel, the decoding of the instruction (iii) is started after the operand of the add instruction (ii) on the buffer memory 10 has been fetched and the operation result has been stored as shown in Fig. 8c. Thus, there is a 4-cycle overhead. On the other hand, in the present invention, since the branch instruction (i) and the add instruction (ii) can be decoded in parallel as shown in Fig. 8d, the overhead to the instruction (iii) can be shortened to three cycles.

As seen from Figs. 8a - 8d, the instruction execution speed is increased depending only on the combination of the instructions extracted into the instruction registers 1 and 2 in parallel without regard to the sequence of the two instructions. In addition, there is no need to increase the hardware of the operation unit.

In order to parallelly decode the instructions only by the combination of the instructions, it is necessary that the instruction registers 1 and 2 and the address generation circuits 8 and 9 or the buffer memories 10 and 11 can be interconnected in any way. In the processor shown in Fig. 1, the selector 4 attains this function. The address generation circuits 8 and 9 and the buffer memories 10 and 11 are correlated, respectively, the output of the address generation circuit 8 is sent only to the buffer memory 10, and the output of the address generation circuit 9 is sent only to the buffer memory 11. When the instruction extracted into the instruction register 1 refers the buffer memory 10, the selector 4 controls such that the address is calculated by the address generation circuit 8, and when the instruction refers the buffer memory 11, the selector 4 controls such that the address is calculated by the address generation circuit 9. The same is true for the instruction extracted into the instruction register 2. Another processor for attaining the above function is shown in Fig. 9, in which the instruction registers 1 and 2 are permanently correlated to the address generation circuits 8 and 9, respectively. The address of the instruction extracted into the instruc-

tion register 1 is always calculated by the address generation circuit 8, and the address of the instruction extracted into the instruction register 2 is always calculated by the address generation circuit 9. When the buffer memory 11 is to be referred by the address calculated by the address generation circuit 8, the signal line 128 is selected by the selector 802 and the instruction is fetched. When the buffer memory 10 is to be referred by the address calculated by the address generation circuit 9, the signal line 131 is selected by the selector 801 and the operand is fetched. The overall configuration of the processor is substantially same as that of Fig. 1, and only the different functions are explained below.

When a 6-byte length instruction is extracted into the instruction register 1, the selector 800 switches the address calculation of the first operand and the address calculation of the second operand. Detail of the selector 800 is shown in Fig. 10. The input signal line 265 is same as that shown in Fig. 5. When the signal line 265 is set, the signal lines 109 and 110 are selected to calculate the address of the second operand. The signal line 265 is set only when the second operand of the 6-byte length instruction is to be operated. When the signal line 265 is not set, the signal lines 107 and 108 are selected. The selectors 801 and 802 select the addresses to be used for the memories. When the instruction in the instruction register 2 refers the buffer memory 10, the signal line 264 is set, and the signal line 131 is selected by the selector 801. When the instruction in the instruction register 1 refers the buffer memory 11, the signal line 804 is selected, and the signal line 128 is set by the selector 802. When the signal lines 264 and 804 are not set, the selector 801 selects the signal line 128 and the selector 802 selects the signal line 131. As shown in Fig. 5, when the first instruction does not refer the buffer memory 10 and the second instruction refers the buffer memory 10 and the second instruction is valid (the signal line 250 is set), the signal line 264 is set. As shown in Fig. 11, when the first instruction refers the buffer memory 11 and the first instruction is valid (the signal line 153 is set), the signal line 804 is set.

As seen from the above description, the processor shown in Fig. 9 can parallelly decode the instructions depending on the combination of the instructions extracted into the instruction registers 1 and 2 and parallelly refer the buffer memories 10 and 11. As a result, the same effect as that shown in Fig. 8 is attained.

In accordance with the present invention, two instructions can be parallelly extracted and they can be parallelly decoded depending on the combination of the instruction. The fetching of the operand of the succeeding instruction and the fetching of the branch-to instruction of the branch instruction can be performed one cycle faster than a case where the parallel decoding is not effected. As a result, the instruction execution time is shortened by one cycle, and the execution time of the succeeding instruction which utilizes the result of the preceding instruction and the execution time of the branch-to instruction are shortened.

Claims

1. A pipeline data processor for executing a plurality of instructions in parallel, comprising

a main storage (13),

buffer means (10, 11) connected to said main storage (13) for holding operands and instructions,

operation means (12) connected to said main storage (13) for performing operations in response to a plurality of instructions other than branch instructions,

resource means responsive to a branch instruction for retrieving a branch-to instruction from said main storage (13),

a pair of instruction registers (1, 2),

set means (3) for retrieving from said main storage (13) and setting into said pair of instruction registers (1, 2) a first instruction to be executed next and a second instruction to be executed next to the first instruction,

first and second address generators (8, 9) connected to said instruction registers (1, 2) for calculating addresses in response to said first and second instructions, conflict detection means (5) for detecting whether the first instruction changes the register used in the address generation of the second instruction, and

detection means (6; 803) for detecting whether the first instruction in said pair of instruction registers (1, 2) is an instruction other than a branch instruction and the second instruction is a branch instruction,

characterized in

that said buffer means includes a first buffer (10) adapted to hold an operand and a second buffer (11) adapted to hold an instruction, and

that selector means (4; 801, 802) is provided which is so controlled that, when the result of detection by said detection means (6; 803) is affirmative, it causes the address relating to the second instruction to be calculated and applied to said second buffer (11) simultaneously with the address relating to the operand of the first instruction being calculated and applied to said first buffer (10), otherwise causes only the address relating to the first instruction to be calculated and applied to respectively said second buffer (11) or said first buffer according to whether said first instruction is a branch instruction or not.

2. The data processor of claim 1,
wherein said first address generator (8) is fixedly connected to said first buffer (10) and said second address generator (9) is fixedly connected to said second buffer (11), and
wherein said selector means includes a common selector (4) for selectively interconnecting said pair of instruction registers (1, 2) and said first and second address generators (8, 9).

3. The data processor of claim 1, wherein said selector means includes a first selector (801) for selectively connecting said first and second address generators (8, 9) to said first buffer (10) and a second selector (802) for selectively connecting said first and second address generators (8, 9) to said second buffer (11).

Patentansprüche

1. Datenprozessor mit Pipeline-Struktur zur parallelen Ausführung mehrerer Befehle, umfassend
einen Hauptspeicher (13),
eine mit dem Hauptspeicher (13) verbundene Puffereinrichtung (10, 11) zur Aufnahme von Operanden und Befehlen,
eine mit dem Hauptspeicher (13) verbundene Operationseinrichtung (12) zur Durchführung von Operationen entsprechend mehreren Befehlen, die keine Verzweigungsbefehle sind,
eine auf einen Verzweigungsbefehl ansprechende Betriebsmitteleinrichtung zum Abholen eines Befehls, der eine Verzweigung zu einem bestimmten Ziel angibt, aus dem Hauptspeicher (13),
ein Paar von Befehlsregistern (1, 2),
eine Eingabeeinrichtung (3) zum Abholen eines als nächsten auszuführenden ersten Befehls und eines im Anschluß an diesen auszuführenden zweiten Befehls aus dem Hauptspeicher (13) und zum Eingeben dieser beiden Befehle in das Befehlsregisterpaar (1, 2),
einen ersten und einen zweiten Adressengenerator (8, 9), die mit den Befehlsregistern (1, 2) verbunden sind, zur Berechnung von Adressen entsprechend dem ersten und dem zweiten Befehl,
eine Konflikt-Erkennungseinrichtung (5) zur Erkennung, ob der erste Befehl das bei der Adressenerzeugung des zweiten Befehls verwendete Register ändert, und
eine Erkennungseinrichtung (6; 803) zur Erkennung, ob der erste Befehl in dem Befehlsregisterpaar (1, 2) ein von einem Verzweigungsbefehl verschiedener Befehl und der zweite Befehl ein Verzweigungsbefehl ist,
dadurch gekennzeichnet,
daß die Puffereinrichtung einen ersten Puffer (10) zur Aufnahme eines Operanden und einen zweiten Puffer (11) zur Aufnahme eines Befehls aufweist, und
daß eine Wähleinrichtung (4; 801, 802) vorgesehen ist, die so gesteuert ist, daß dann, wenn das Erkennungsergebnis der Erkennungseinrichtung (6; 803) positiv ist, bewirkt, daß die auf den zweiten Befehl bezogene Adresse berechnet und dem zweiten Puffer (11) zugeführt und gleichzeitig die auf den Operanden des ersten Befehls bezogene Adresse berechnet und dem ersten Puffer (10) zugeführt wird, während sie sonst bewirkt, daß nur die auf den ersten Befehl bezogene Adresse berechnet und in Abhängigkeit davon, ob dieser ein Verzweigungsbefehl ist oder nicht, dem zweiten Puffer (11) bzw. dem ersten Puffer zuführt wird.

2. Datenprozessor nach Anspruch 1,
wobei der erste Adressengenerator (8) mit dem ersten Puffer (10) und der zweite Adressengenerator (9) mit dem zweiten Puffer (11) fest verbunden ist, und
wobei die Wähleinrichtung einen gemeinsamen Wähler (4) zum selektiven Verbinden des Befehlsregisterpaars (1, 2) mit dem ersten und zweiten Adressengenerator (8, 9) aufweist.

3. Datenprozessor nach Anspruch 1, wobei die Wähleinrichtung einen ersten Wähler (801) zum selektiven Verbinden des ersten und des zweiten Adressengenerators (8, 9) mit dem ersten Puffer (10) und einen zweiten Wähler (802) zur selektiven Verbinden des ersten und des zweiten Adressengenerators (8, 9) mit dem zweiten Puffer (11) aufweist.

Revendications

1. Processeur de données en pipeline pour exécuter une pluralité d'instructions en parallèle, comprenant
une mémoire centrale (13),
un moyen formant tampon (10, 11) connecté à ladite mémoire centrale (13) pour contenir des opérandes et des instructions,
un moyen d'exploitation (12) connecté à ladite mémoire centrale (13) pour effectuer des opérations en réponse à une pluralité d'instructions autres que des instructions de branchement,
un moyen formant ressource réagissant à une instruction de branchement pour extraire de ladite mémoire centrale (13) une instruction de branchement,
une paire de registres (1, 2) d'instructions,

un moyen d'affectation (3) pour extraire de ladite mémoire centrale (13) et placer dans ladite paire de registres (1, 2) d'instructions une première instruction devant être exécutée comme prochaine instruction et une seconde instruction devant être exécutée après la première instruction,

un premier et un second générateurs (8, 9) d'adresses connectés auxdits registres (1, 2) d'instructions pour calculer des adresses en réponse auxdites première et seconde instructions,

un moyen (5) de détection de conflit pour détecter si la première instruction modifie le registre utilisé lors de la génération d'adresse de la seconde instruction, et

un moyen de détection (6; 803) pour détecter si la première instruction dans ladite paire de registres (1, 2) d'instructions est une instruction autre qu'une instruction de branchement et si la seconde instruction est une instruction de branchement,

caractérisé en ce que

ledit moyen formant tampon (10) comporte un premier tampon (10) conçu pour contenir un opérande et un second tampon (11) conçu pour contenir une instruction, et

en ce qu'il est prévu un moyen formant sélecteur (4; 801, 802) qui est commandé de façon que, lorsque le résultat de détection par ledit moyen de détection (6; 803) est affirmatif, il amène l'adresse relative à la seconde instruction à être calculée et appliquée audit second tampon (11) en même temps que l'adresse relative à l'opérande de la première instruction est calculée et appliquée audit premier tampon (10), autrement il amène seulement l'adresse relative à la première instruction à être calculée et appliquée respectivement audit second tampon (11) ou audit premier tampon selon que ladite première instruction est ou n'est pas une instruction de branchement.

2. Processeur de données selon la revendication 1,

dans lequel ledit premier générateur (8) d'adresse est connecté de manière fixe audit premier tampon (10) et ledit second générateur (9) d'adresse est connecté de manière fixe audit second tampon (11), et

dans lequel ledit moyen formant sélecteur comporte un sélecteur commun (4) pour interconnecter de manière sélective ladite paire de registres (1, 2) d'instructions et lesdits premier et second générateurs (8, 9) d'adresses.

3. Processeur de données selon la revendication 1, dans lequel ledit moyen formant sélecteur comporte un premier sélecteur (801) pour connecter sélectivement lesdits premier et second générateurs (8, 9) d'adresses audit premier tampon (10) et un second sélecteur (802) pour connecter sélectivement lesdits premier et second générateurs (8, 9) d'adresses audit second tampon (11).

FIG. 1

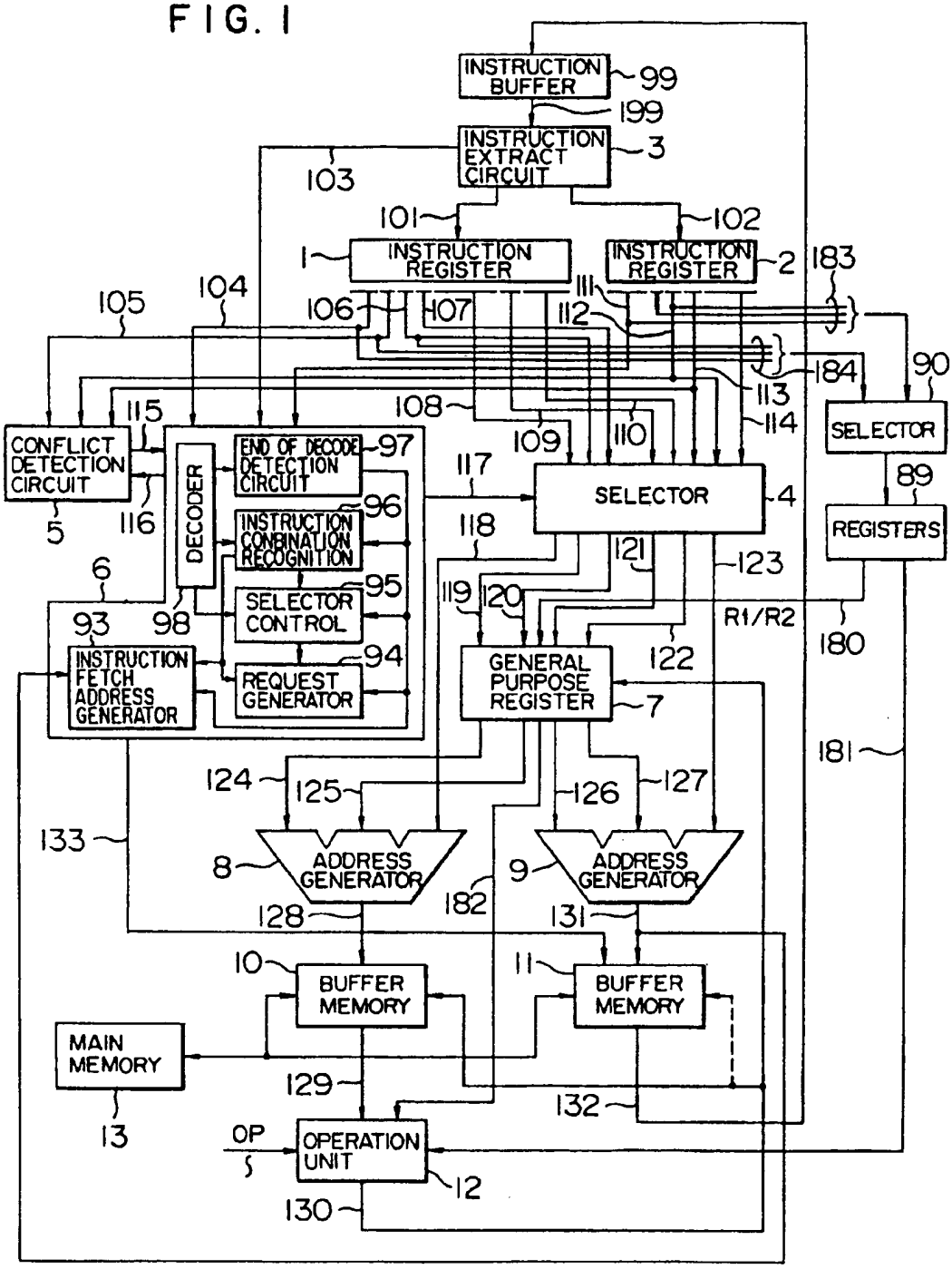
INSTRUCTION BUFFER
99
199
INSTRUCTION EXTRACT CIRCUIT
3
103
101
102
1
INSTRUCTION REGISTER
INSTRUCTION REGISTER
2
183
105
104
106
107
111
112
184
90
113
108
115
109
110
114
SELECTOR
CONFLICT DETECTION CIRCUIT
DECODER
END OF DECODE DETECTION CIRCUIT
97
117
SELECTOR
4
89
5
116
INSTRUCTION COMBINATION RECOGNITION
96
118
REGISTERS
6
121
123
SELECTOR CONTROL
95
119
120
93
98
R1/R2
180
INSTRUCTION FETCH ADDRESS GENERATOR
REQUEST GENERATOR
94
GENERAL PURPOSE REGISTER
122
7
181
124
125
126
127
133
ADDRESS GENERATOR
ADDRESS GENERATOR
8
9
182
128
131
10
11
BUFFER MEMORY
BUFFER MEMORY
MAIN MEMORY
129
132
13
OP
OPERATION UNIT
12
130

FIG. 2a

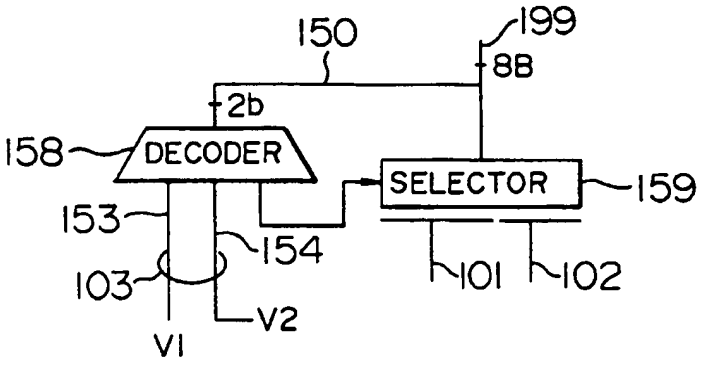

FIG. 2b

FIRST TWO BITS	INPUT 101 TO FIRST INSTRUCTION REGISTER	V1	INPUT 102 TO SECOND INSTRUCTION REGISTER	V2
00	1 2	1	3 4 5 6	1
01 10	1 2 3 4	1	5 6 7 8	1
11	1 2 3 4 5 6	1		0

FIG. 3a

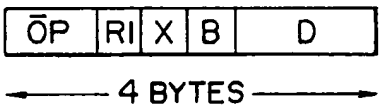

FIG. 3b

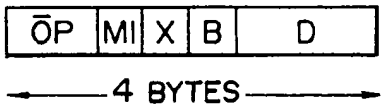

FIG. 3c

FIG. 4

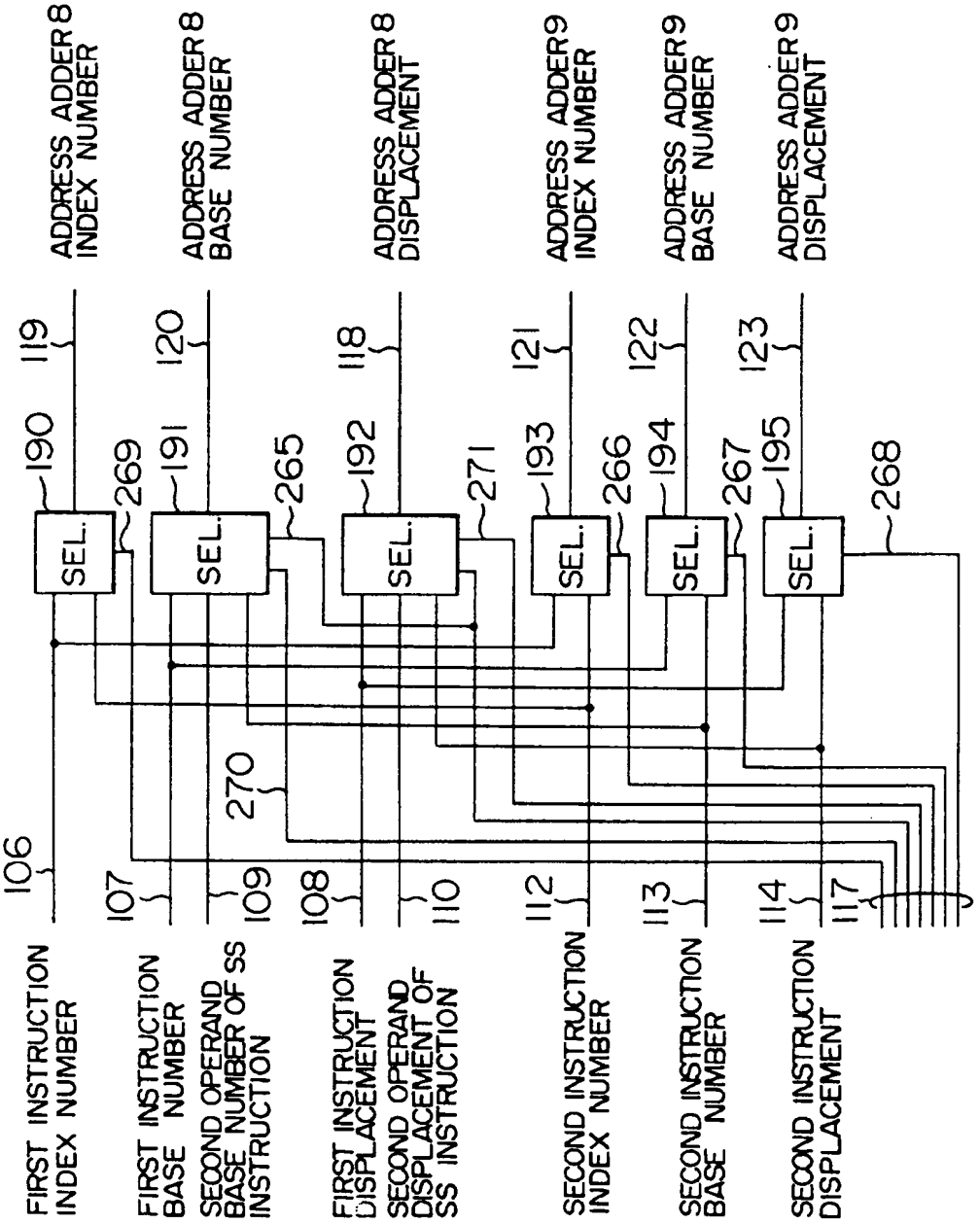
ADDRESS ADDER 8 INDEX NUMBER
ADDRESS ADDER 8 BASE NUMBER
ADDRESS ADDER 8 DISPLACEMENT
ADDRESS ADDER 9 INDEX NUMBER
ADDRESS ADDER 9 BASE NUMBER
ADDRESS ADDER 9 DISPLACEMENT
SEL.
190
191
192
193
194
195
269
265
271
266
267
268
119
120
118
121
122
123
106
107
109
108
110
112
113
114
117
270
FIRST INSTRUCTION INDEX NUMBER
FIRST INSTRUCTION BASE NUMBER
SECOND OPERAND BASE NUMBER OF SS INSTRUCTION
FIRST INSTRUCTION DISPLACEMENT
SECOND OPERAND DISPLACEMENT OF SS INSTRUCTION
SECOND INSTRUCTION INDEX NUMBER
SECOND INSTRUCTION BASE NUMBER
SECOND INSTRUCTION DISPLACEMENT

FIG.5

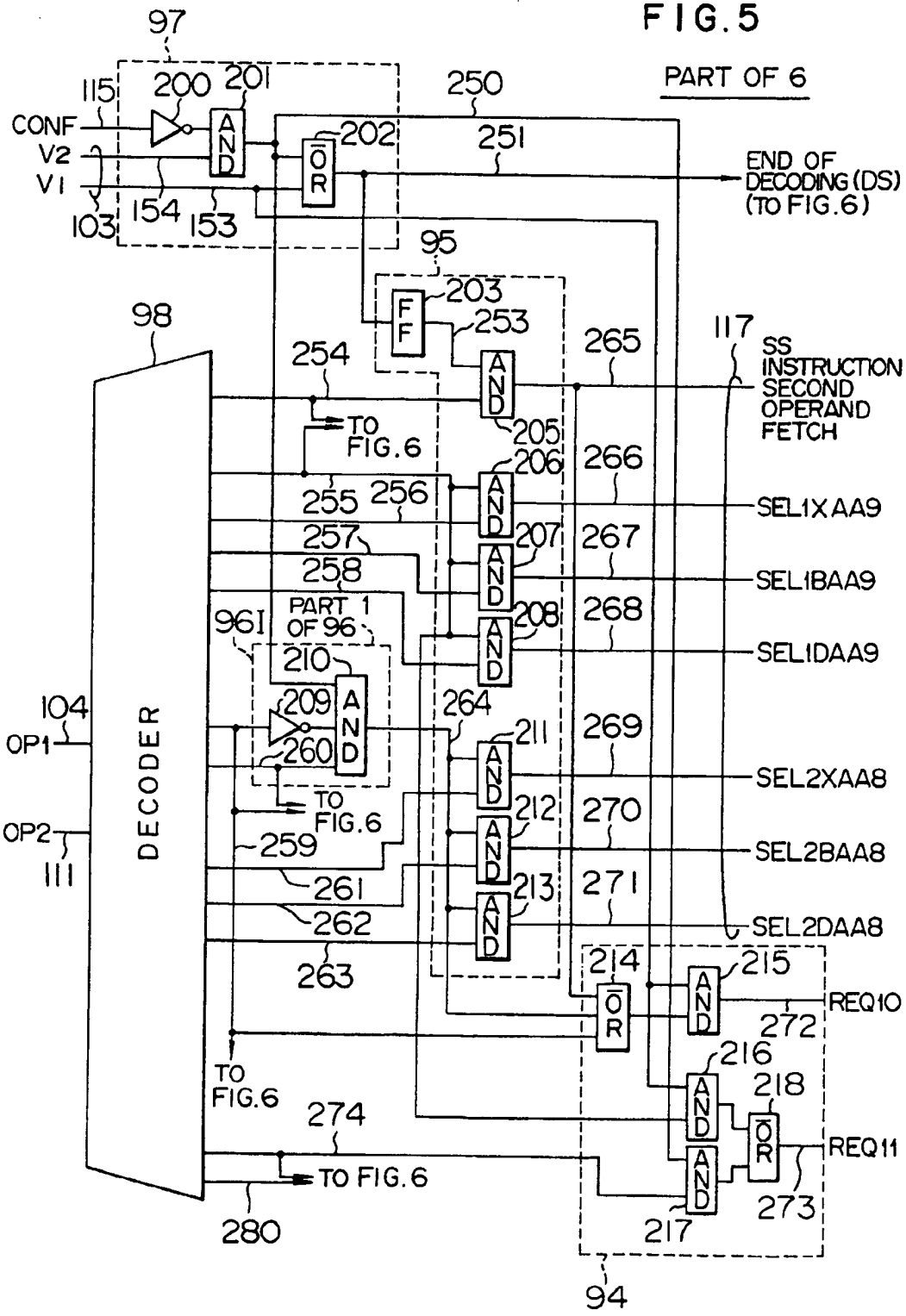
97
115
200
201
250
PART OF 6
CONF
V2
V1
AND
OR
202
251
END OF DECODING (DS) (TO FIG.6)
103
154
153
95
FF
203
253
98
254
265
117
SS INSTRUCTION SECOND OPERAND FETCH
TO FIG.6
205
206
266
255
256
SEL1XAA9
257
207
267
258
SEL1BAA9
PART 1 OF 96
961
208
268
SEL1DAA9
210
209
264
211
269
104
OP1
260
SEL2XAA8
DECODER
TO FIG.6
212
270
OP2
259
SEL2BAA8
111
261
213
271
262
SEL2DAA8
263
214
215
REQ10
272
216
218
TO FIG.6
274
REQ11
TO FIG.6
273
280
217
94

FIG. 6

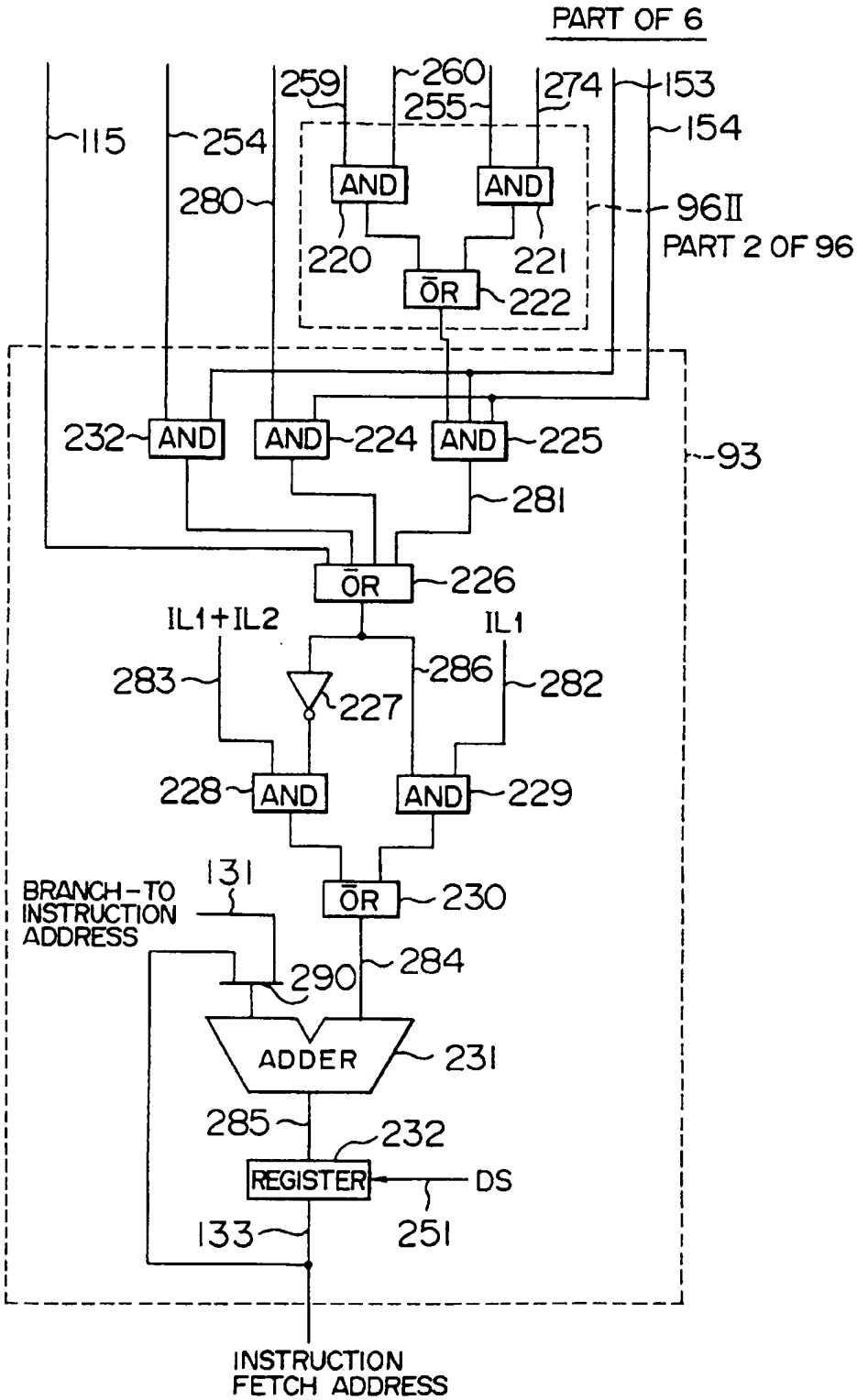
PART OF 6
259
260
255
274
153
154
115
254
AND
AND
280
96II
PART 2 OF 96
220
221
ŌR
222
232
AND
AND
224
AND
225
93
281
ŌR
226
IL1+IL2
IL1
283
286
282
227
228
AND
AND
229
131
BRANCH-TO INSTRUCTION ADDRESS
ŌR
230
290
284
ADDER
231
285
232
REGISTER
DS
133
251
INSTRUCTION FETCH ADDRESS

FIG. 7

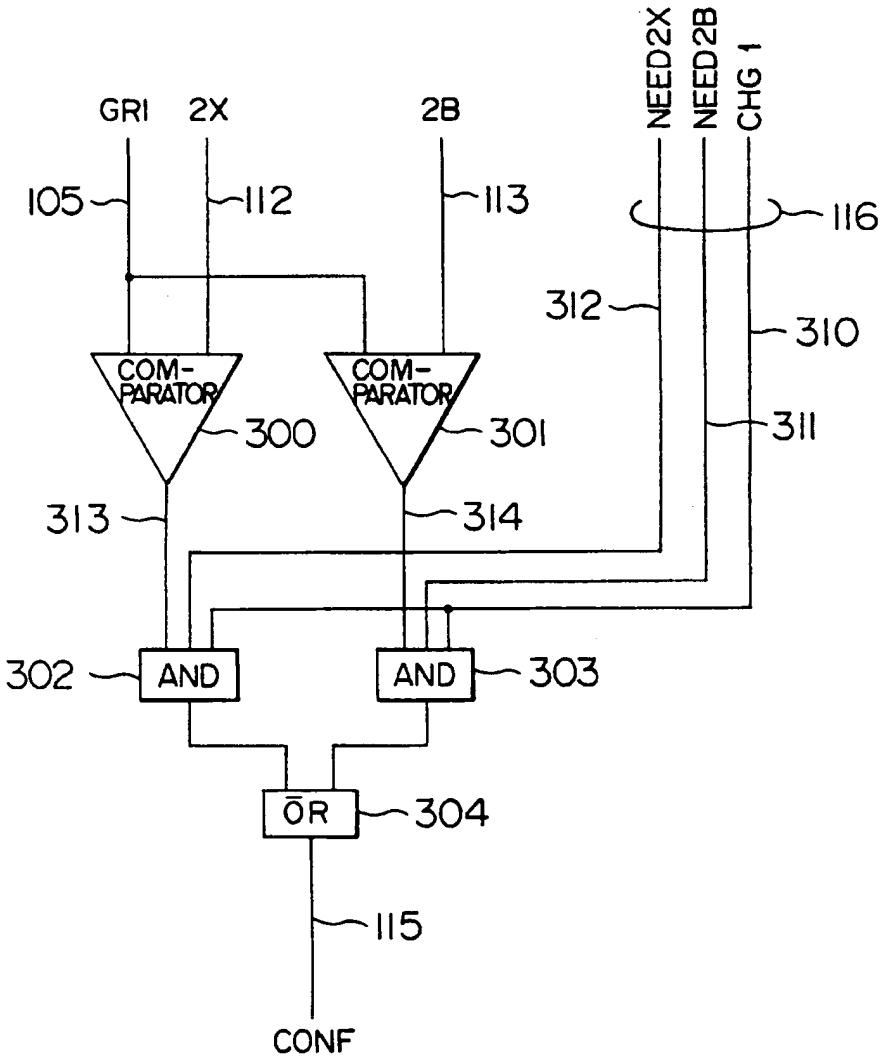
GRI
2X
2B
NEED2X
NEED2B
CHG 1
105
112
113
116
312
310
COM-PARATOR
300
COM-PARATOR
301
311
313
314
302
AND
AND
303
OR
304
115
CONF

FIG.8a

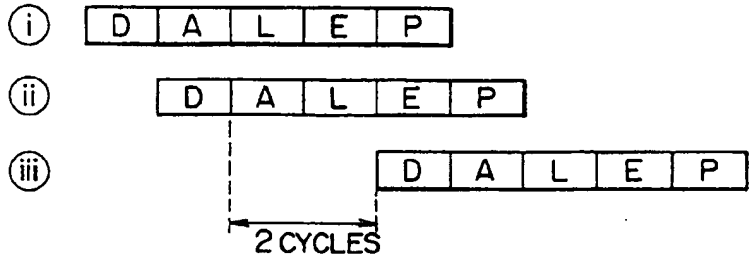

FIG.8b

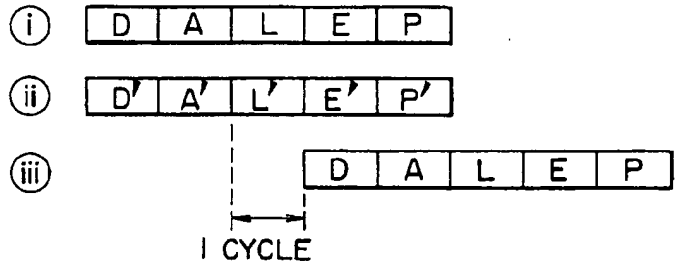

FIG.8c

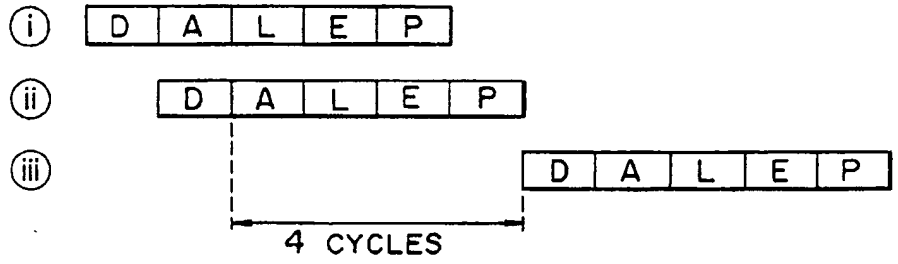

FIG.8d

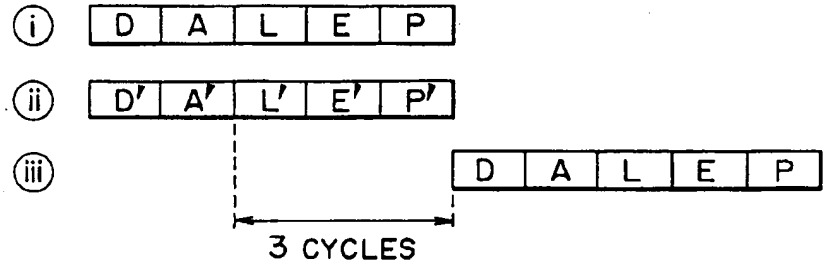

FIG. 9

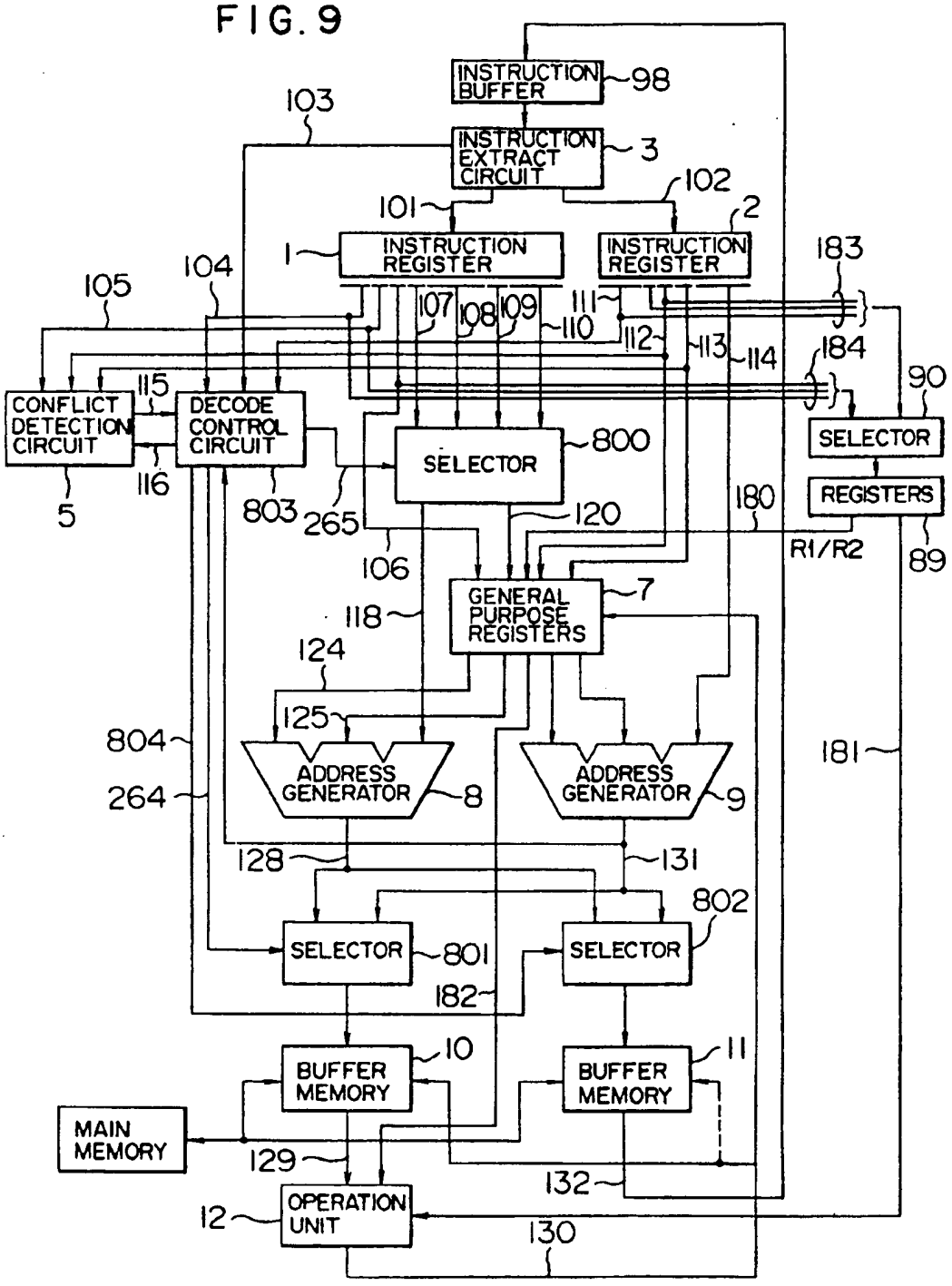
INSTRUCTION BUFFER
98
103
INSTRUCTION EXTRACT CIRCUIT
3
102
101
2
1
INSTRUCTION REGISTER
INSTRUCTION REGISTER
183
104
105
107
108
109
110
111
112
113
114
184
90
115
CONFLICT DETECTION CIRCUIT
DECODE CONTROL CIRCUIT
SELECTOR
800
SELECTOR
REGISTERS
5
116
803
265
120
180
R1/R2
89
106
GENERAL PURPOSE REGISTERS
7
118
124
125
804
264
ADDRESS GENERATOR
8
ADDRESS GENERATOR
9
181
128
131
802
SELECTOR
801
SELECTOR
182
BUFFER MEMORY
10
BUFFER MEMORY
11
MAIN MEMORY
129
132
12
OPERATION UNIT
130

FIG. 10

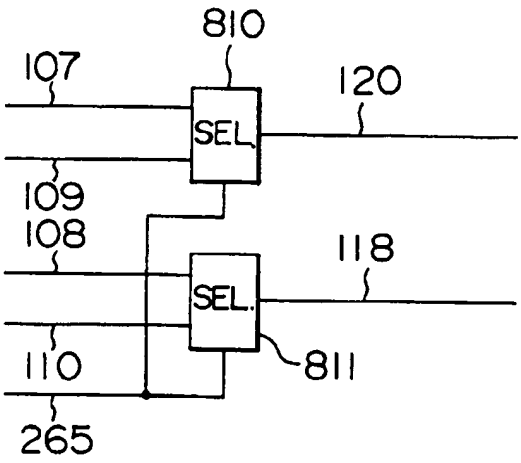

FIG. 11